Write C++ code for a loop that simultaneously computes both the maximum and minimum of a vector.

```
for (int i = 0; i < a.size(); i++) {
  if (min > a[i]) min = a[i];
  if (max < a[i]) max = a[i];
}
```

Write a loop that reads ten numbers and a second loop that displays them in the opposite order from which they were entered.

```
vector <int> holding;
        for (int i = 0; i < 10; i++)
        {
                cout << "please enter a number" << endl;
                int temp = 0;
                cin >> temp;
                holding.push_back(temp);
        }
        for (int j = 9; j >= 0; j--)
        {
                cout << holding[j] << " ";
        }
        cout << endl;
```

Exercise R6.9. Give an example of
a. A useful function that has a vector of integers as a value parameter.
b. A useful function that has a vector of integers as a reference parameter.
c. A useful function that has a vector of integers as a return value.

6.9.a:
void parkedCar(vector <int> spots, int ticket);
parkedCar returns information on a car that is parked in a specific spot in a parking lot, say for a dealership or in a parking garage. The “ticket” variable would hold the number spot that the car is in, used as an index number for the vector “spots”, and return the make, model, and price or, if its for a parking garage, how long the car has been there and how much money is owed.

6.9.b:
Void setSpot(vector <int> & spots, int ticket, car newCar);
setSpot takes a vector spots and finds an index of the vector using ticket, and changes the value of the element at the vector from what it was to newCar. This could be for a car dealership that got new cars to add to their lot.

6.9.c:
vector <int> getColor(vector <car> cars, string red);

getColor takes a vector of car objects and a string representing a color of car and returns a vector of the numbered spots in a lot of cars that have a car of the color passed to the function.

Exercise R6.13. Suppose v is a sorted vector of employees. Give pseudocode that describes how a new employee can be inserted in its proper position so that the resulting vector stays sorted.

(assuming that the operators have been set up for the employee objects)

For a loop going from zero to the number of elements in the vector minus one
If the employee to be inserted is greater than the element that the loop is at currently and the employee is less than the element that is found at the index of the number the loop is at plus one, then the employee is inserted at the current index of the vector and the rest of the vector is moved up one.
otherwise the loop just continues

Exercise R6.15. How do you perform the following tasks with vectors in C++?
a. Test that two vectors contain the same elements in the same order.
b. Copy one vector to another. (Hint: You may copy more than one element at a time.)
c. Fill a vector with zeroes, overwriting all elements in it.
d. Remove all elements from a vector. (Hint: You need not remove them one by one.)

R6.15.a: (given that the vectors contain variables that are comparable with operators) after checking to make sure the vectors have the same number of elements, a for loop is set up to go from 0 to the number of elements, and in the loop the elements at each index of the vectors are compared to see if they are identical. If they are not identical at any point a bool is set to false.

R6.15.b:

```
Vector <int> temp;
Vector <int> tempTwo;
for(int i = 0; i < 10 ; i++)
{
        temp.push_back(0);
        tempTwo.push_back(1);
}
for(int j = 0; j < 10;j++)
{
```

```
        temp.push_back(tempTwo[j]);
}
```

R6.15.c:

```
Vector <int> zeros;
for(int i = 0; i < 10 ; i++)
{
        zeros.push_back(0);
}
for(int j = 0; j < 10;j++)
{
        Zeros[j] = j;
}
```

R6.15.d:

```
vecortName.clear();
```